Actualización de Replicas utilizando Agentes Móviles en Entornos Distribuidos sin Conexión Permanente: una experiencia.

Lic. Ivana Miaton[1] Lic. Rodolfo Bertone[2]

Lic. Patricia Pesado[3] Ing. Armando De Giusti[4]

L.I.D.I. (Laboratorio de Investigación y

Desarrollo en Informática)[5]

Facultad de Informática

UNLP

Resumen

El paradigma de agentes móviles permite acceder, recolectar o actualizar información desde cualquier estación de trabajo dentro de una red heterogénea de computadoras. De esta forma, el código ejecutable de un agente móvil puede migrar entre los puestos de trabajo.

Mantener una BD distribuida actualizada en cualquier lugar, en cualquier momento y de cualquier forma mediante transacciones tiene un comportamiento inestable con cargas de trabajo que varían de acuerdo a la utilización de la BD y de la red de computadoras. Si bien existen variantes para actualización de réplicas, como por ejemplo un esquema master-slave, que ayuda a solucionar algunos de estos problemas, se están desarrollando algoritmos que utilizan agentes móviles para manipular las transacciones que realizan estos movimientos sobre la BD.

Este trabajo presenta el primer estudio realizado para actualizar réplicas de una BDD mediante el uso de agentes móviles, generados por estaciones de trabajo, las cuales eventualmente pueden permanecer mucho tiempo desconectadas del sistema.

Palabras Clave

Sistemas Distribuidos – Agentes Móviles – Replicación de Información – Actualización de Réplicas

[1] imiaton@lidi.info.unlp.edu.ar Ayudante Diplomada Dedicación Semi Exclusiva – Facultad de Informática - UNLP
[2] rbertone@lidi.info.unlp.edu.ar Profesor Adjunto Dedicación Exclusiva - Facultad de Informática - UNLP
[3] ppesado@lidi.info.unlp.edu.ar Profesor Titular Dedicación Exclusiva - Facultad de Informática - UNLP
[4] degiusti@lidi.info.unlp.edu.ar Profesor Titular Dedicación Exclusiva – Facultad de Informática – UNLP
Investigador Principal CONICET
[5] 50 y 115 (1900) La Plata – Buenos Aires – Argentina TE-FAX: + 54 221 4227707

Introducción

Agentes Móviles

Los agentes móviles son procesos de software capaces de viajar entre nodos de redes heterogéneas de computadoras, interactuando con ellos, recuperando o actualizando información en favor de su propietario y regresando a la localidad originaria con los resultados obtenidos. La movilidad no es característica suficiente para la agenticidad, los agentes móviles son autónomos (pueden definir la ruta a viajar) y cooperativos (pueden intercambiar datos e información con otros agentes). [PEREZ 98]

La principal hipótesis de los agentes móviles es que no necesitan ser estacionarios. Proveen ventajas como:

- Reducción de costos de comunicación
- Utilización de recursos no propios, en caso que los recursos locales resultan limitados.
- Fácil coordinación
- Comunicación asíncrona
- Proveen un ambiente natural desarrollado para la implementación de servicios en un "mercado libre"
- Una arquitectura flexible de computación distribuida
- Representan una oportunidad distinta para abordar el diseño de procesos en general.

Los agentes de red o agentes móviles han sido desarrollados como un método alternativo para el paradigma de Llamados a Procedimientos Remotos (RPC). La característica principal de RPC es que, en cada interacción entre el cliente y el servidor se entablan dos actos de comunicación, uno para enviarle al servidor la petición y los argumentos correspondientes y otra para enviar los resultados de la petición. Este tipo de interacción puede incrementar dramáticamente él trafico de la red si el número de clientes y/o el número de peticiones de servicios se incrementan.

Una alternativa a RPC es la programación Remota (RP) también referenciado como modelo basado en agentes móviles [SÁNCHEZ 97]. En este tipo de comunicación al enviar el cliente A una petición para la ejecución de un servicio no solamente manda los argumentos necesarios para la ejecución de tal servicio, sino que manda el procedimiento requerido para su ejecución. Cada mensaje que la red transporta contiene un procedimiento que la computadora receptora ejecuta y los datos (argumentos) necesarios para su ejecución. Estos datos reflejan el estado actual del procedimiento. El procedimiento comienza su ejecución en el cliente pero continua en el servidor. El procedimiento y su estado son llamados un agente móvil. La figura 1 presenta el esquema que representa a un agente móvil con el paradigma RP.

La primera ventaja de RP es el desempeño. Cuando una computadora cliente tiene trabajo que necesite ejecutarse en un servidor remoto, puede enviar el trabajo y supervisarlo localmente usando un agente, en vez de estar enviando continuamente instrucciones sobre la red. La segunda ventaja de RP es la personalización. Un agente móvil permite que el cliente personalice la funcionalidad del servidor. Pueden enviarse nuevos procedimientos con poco esfuerzo. RP puede ayudar para automatizar procesos de instalación. Se puede codificar en un programa un proceso de instalación y transferirlo a un conjunto de nodos de la red. En cada nodo el programa puede analizar las características de la plataforma local de hardware / software y ejecutar la configuración correcta. RP ayuda a incrementar la flexibilidad del servidor, manteniendo limitado el tamaño y la

complejidad del mismo. Cada cliente es responsable de la correcta especificación del servicio que necesita y debe ser descripto en el código enviado al servidor remoto.

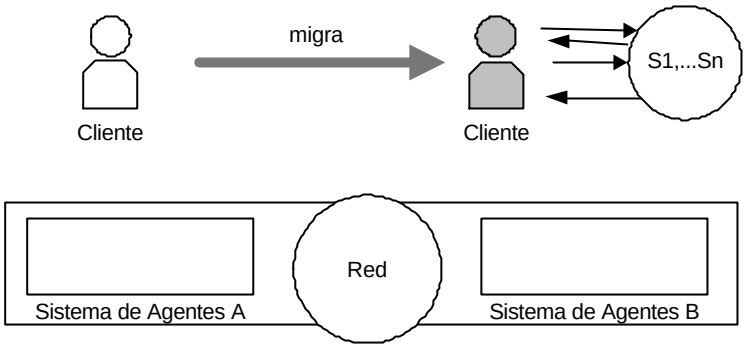

Figura 1 Paradigma RP.

En resumen se puede decir que un agente móvil es un programa o proceso con las siguientes características:

1. Es autónomo o semi-autónomo de manera que él decide cómo, cuándo y a dónde migrar.
2. Esta orientado a ejecutar tareas, a veces en nombre del usuario y otras basándose en los cambios de su ambiente.
3. Se envía como objeto, a través de plataformas conservando además de su código, los datos y su estado de ejecución.
4. Es asíncrono, debido a que tiene su propio proceso o hilo de ejecución. Por tanto, el agente se ejecuta asincrónicamente respecto a los otros procesos que se estén ejecutando en el nodo.
5. Es capaz de comunicarse con su dueño, con otros agentes y con el medio.
6. Puede operar sin conexión, es decir, que puede ejecutar sus tareas aún cuando la conexión a red no este funcionando; si el agente necesita trasladarse y la red no esta activa, el agente puede esperar o desactivarse hasta que la conexión se restablezca.
7. Puede suspender su ejecución, transportarse a otro host y reanudar su ejecución desde el punto en el cual se suspendió.
8. Es capaz de duplicarse.
9. Puede reaccionar a cambios en su ambiente, modificando su conducta debido a las acciones generadas por otros agentes, debido a su experiencia propia o por la intervención directa del programador o usuario.

Tratamiento de Replicación de datos

Los datos son replicados en múltiples nodos de una red por cuestiones de performance y disponibilidad. La replicación Eager [BURETTA 97] mantiene todas las copias exactamente sincronizadas en todos los nodos modificando todas las réplicas como parte de una transacción atómica. Esta replicación presenta características de ejecución serializable, pero no evita anomalías

de concurrencia. El costo de este esquema es en performance, incrementando el tiempo de respuesta de una transacción. [GRAY 96]

Este esquema no es una opción posible si se piensa en aplicaciones que utilicen agentes móviles donde los nodos pueden permanecer desconectados mucho tiempo. Las aplicaciones con agentes móviles requieren replicación Lazy [LADIN 92] que propagan asincrónicamente las modificaciones a las réplicas en otros nodos luego que una transacción cometa. Este protocolo presente una serie de inconvenientes que deben tratarse. Entre ellos, el más importante está relacionado con el de control de concurrencia. Se utilizan esquemas de control especiales para detectar comportamiento que no puede serializarse [BERNSTEIN 87].

Los esquemas Eager pueden demorar o abortar una transacción no cometida si considera que se viola la serialización. Esto no puede ocurrir en un esquema Lazy, donde la transacción puede ya haberse cometido, y lo que resta es la actualización de la réplica. En estos casos se debe tener un mecanismo de reconciliación del conflictos.

Actualización de réplicas con Agentes Móviles: El problema

Cuando se estudia el esquema de actualización de réplicas en un entorno distribuido, generalmente se asumen dos métodos: Eager y Lazy. Ahora bien, también existen dos formas para regular la actualización de las réplicas, estas son: Group (cualquier nodo con una copia del dato puede modificarlo) y Master (cada objeto de dato tiene una copia maestra residente en un nodo, solo se puede modificar esta copia, el resto es de solo lectura). [LEN 01]

De acuerdo a la combinación de los métodos de propagación y de regulación de réplicas, se pueden efectuar las modificaciones sobre la BDD. [HOLLIDAY 2002] En el caso que se presenta en este artículo, se disponen de nodos que se denominan base, los cuales se encuentran siempre conectados, y un conjunto de nodos denominados móviles, que generalmente se encuentran desconectados de la red. Esto genera dos aspectos a tener en cuenta:

1- Los nodos móviles generan transacciones de modificación tentativas para objetos que son propiedad de otros nodos (usualmente base).

2- Los nodos móviles ocasionalmente se conectan a los nodos base y proponen la ejecución de transacciones tentativas para modificar la BDD. Las transacciones tentativas pueden ser rechazadas, en el caso de no poder satisfacer la actualización requerida. Estas transacciones rechazadas deben ser tratadas nuevamente por el nodo móvil que la generó, debido a que no llegó a un estado de cometido total.

Cuando cada nodo móvil genera una transacción, utiliza un agente móvil para ejecutarla, debido a que los agentes móviles disponen de las características necesarias para ese efecto.

Como se indicó anteriormente, los esquemas Eager no sirven en este caso, solo los esquemas Lazy son aplicables. Se presenta a continuación un análisis sobre las dos diferentes formas de actualización de réplicas.

Replicación Lazy Group

La replicación Lazy Group permite que cada nodo modifique sus datos locales. Cuando la transacción comete sobre un nodo local, es enviada al resto de los nodos que contienen réplicas, para aplicar las modificaciones pertinentes. En este caso, es posible que dos transacciones generadas por dos nodos diferentes intenten modificar al mismo tiempo un objeto de datos. El mecanismo de replicación debe detectar esta situación y reconciliar las dos transacciones de manera que todas las transacciones producidas sean realizadas.

El mecanismo utilizado para detectar y reconciliar problemas de actualización para un esquema Lazy Group es el hora de entrada [SILVERSCHATZ 98]. Cada objeto lleva la hora de entrada de la actualización más reciente; además, cada transacción que modifica la réplica de la BD lleva el nuevo valor. Este nuevo valor se compara con el valor que ya tiene el objeto para aceptar o no el cambio. En caso de tener el mismo valor la modificación se acepta sin problemas, pero en caso de no coincidir, la transacción es considerada peligrosa, el nodo la rechaza y la envía para reconciliación.

En el caso de computación móvil, donde los nodos no se encuentran conectados *full time*, el problema de actualización utilizando este esquena tiende a complicarse significativamente. Si dos transacciones modifican un objeto de dato en dos nodos diferentes y, una o ambas, están en nodos móviles y, por ende desconectados, necesitan un mecanismo de reconciliación.

¿Cuál será la probabilidad que dos transacciones generadas en nodos móviles colisionen? Este análisis debe realizarse en término del tiempo de desconexión (TD), cantidad de transacciones generadas por un nodo por unidad de tiempo (#TGN), y el número promedio de modificaciones realizadas por un transacción (#MPT).

$$Actualización\ local\ (Al) \gg [\ TD * \#TGN * \#MPT\ ].$$

Estas modificaciones deben efectuarse sobre cada nodo que contenga el objeto replicado. Las modificaciones pendientes que el nodo genera para el resto de la red es aproximadamente *Nodos–1* veces más grande, siendo *Nodos* el número de nodos que contienen los objetos replicados. De aquí que el costo de la modificación, será

$$Actualización\ de\ réplicas\ (Ar) \gg \{[Nodos - 1] * TD * \#TGN * \#MPT\ \}.$$

Si una *Al* y *Ar* se solapan, significa que una actualización local sobre un objeto se realiza simultáneamente con una actualización generada por otro nodo. Se produce una colisión que necesita una reconciliación. La probabilidad de que esto ocurra es:

$$Colisión \gg \frac{Al * Ar}{tamaño\ de\ la\ BD} \gg \frac{Nodos * (TD * \#TGN * \#MPT)^2}{tamaño\ de\ la\ BD}$$

La ecuación anterior es la probabilidad que un nodo móvil necesite el algoritmo de reconciliación, al haber generado una transacción que entró en conflicto.

$$\%Reconciliación\_Lazy\_Group \gg Colisión * \frac{Nodos}{TD}$$

Replicación Lazy Master

La replicación Master asigna un propietario a cada objeto de dato. El propietario tiene el valor correcto para ese objeto en todo momento. Las modificaciones se realizan primero sobre esta copia maestra y luego se propagan al resto de las réplicas.

Una vez que la transacción maestra haya cometido, el nodo que originó la transacción envía la actualización de la réplica a todas las copias secundarias mediante una nueva transacción.

El esquema Lazy Master no es apropiado para aplicaciones móviles, donde el nodo móvil contiene la copia maestra de algún objeto de datos. En esta caso, y debido a la desconexión del nodo, no se puede garantizar la actualización del mismo. Ahora bien, si se trabaja sobre la hipótesis que las localidades móviles se limitan a contener copias *no* maestras de los objetos, dejando solo para nodos bases la ubicación principal de la copia, el esquema de actualización Lazy Master resulta más atractivo.

El estudio se centra, por lo planteado hasta aquí, en analizar y modelizar el comportamiento y la performance del esquema de replicación Lazy, con las variantes Master y Group, utilizando nodos móviles.

Actualización de réplicas con Agentes Móviles: El entorno de trabajo

Para realizar la experiencia de actualización de réplicas mediante agentes móviles se realizaron una serie de suposiciones iniciales, las cuales se describen a continuación:

- Se realizan las primeras pruebas sobre un esquema Lazy-Master.
- Los nodos denominados base contienen copias maestras de los objetos de dato.
- Como corolario de lo anterior, los nodos móviles solamente podrán contener copias secundarias de los objetos de dato.
- Cada nodo de la red posee una copia de Diccionario de Datos del modelo que permite conocer la ubicación de la copia maestra correspondiente a cada objeto de dato.
- Además, el Diccionario de Datos mantiene información sobre la ubicación de las copias secundarias de datos.
- Se propone un esquema libre de fallos. Esto se debe a que los protocolos más usuales que tratan sobre la integridad de la base de datos resultan muy complejos de adaptar para esquemas *wireless*, con nodos usualmente desconectados. Los estudios realizados previamente sobre el comportamiento de los protocolos de cometido de dos y tres fases [MIATON 98] [RUSCUNI 00], junto con sus variantes, resulta inviable para entornos con nodos móviles.

Bajo las precondiciones mencionadas la actualización de las réplicas se realizará mediante la utilización de agentes móviles, generados tanto por nodos móviles como por nodos base. Se describen a continuación las diferentes posibilidades.

Agentes móviles generados por nodos móviles

Un nodo móvil genera un agente móvil para intentar realizar una modificación de una copia maestra de algún objeto de dato de la BD. De esta forma, en el momento en que el nodo móvil se conecta a la red envía un agente hacia el nodo base donde se encuentra la copia maestra del objeto que desea actualizar, llevando con sí la transacción a efectivizar.

Cuando el agente llega al nodo base, se encarga de hacer ejecutar la transacción. Como se supone un modelo libre de fallos, la transacción podrá o no tener éxito pero uno de esto dos resultados está garantizado. En caso de alcanzar el estado de cometido, el nodo base disparará un nuevo agente que se encargue de llevar esta actualización hacia cada copia secundaria del objeto de dato modificado. Este comportamiento se describe a continuación.

Agentes móviles generados por nodos base

Los nodos base generan agentes móviles para actualizar las copias secundarias de los objetos que, en ese nodo base, figuren como maestros. Las posibilidades para la generación de agentes móviles son dos:

- Se genera un agente que contenga la transacción que modificará cada copia. Luego se clona dicho agente tantas veces como copias secundarias posea ese objeto de dato. Y, posteriormente, se envía cada clon al nodo adecuado. (Figura 2)

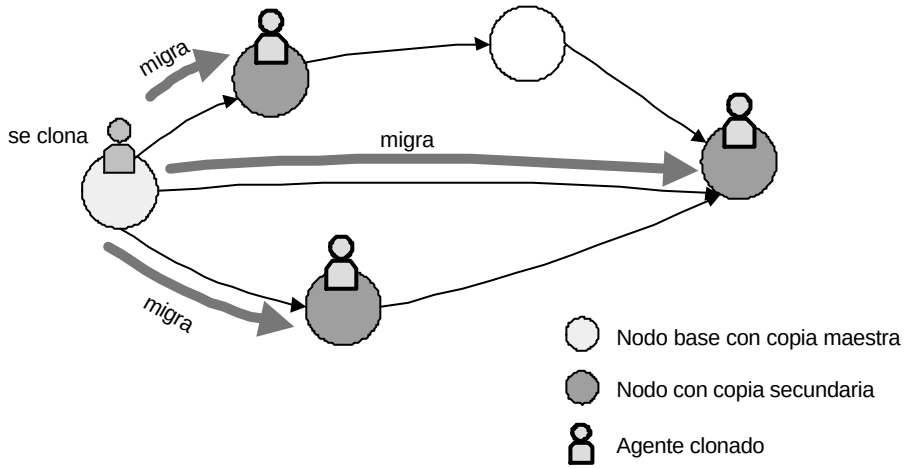

Figura 2: agente que se clona para modificar copias secundarias

- Se genera un solo agente con la transacción que modificará cada copia, y, posteriormente, se asigna al agente un itinerario que debe cubrir. En dicho itinerario se definen los diferentes nodos que debe "visitar" el agente (Figura 3).

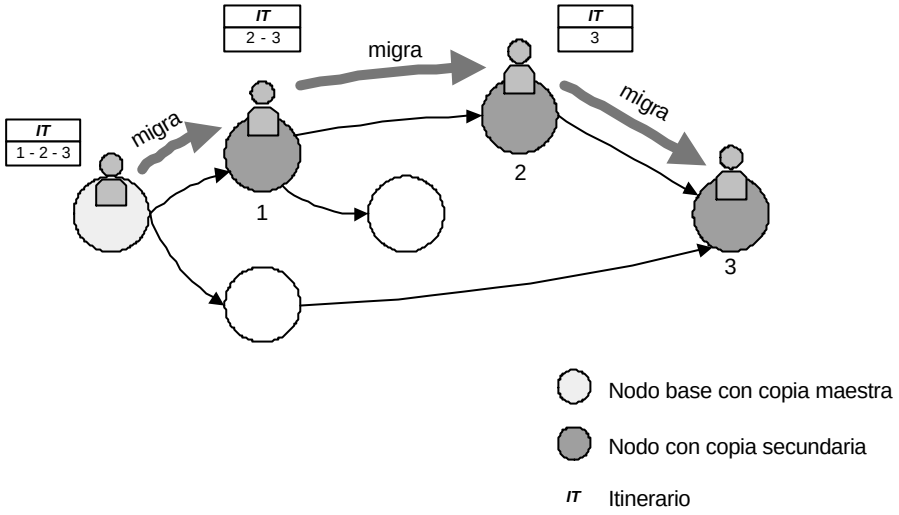

Figura 3: agente con itinerario para modificar copias secundarias

En cualquiera de las dos alternativas planteadas hay que tener en cuenta que los agentes generados por un nodo base, con el fin de actualizar copias de objetos de datos, pueden dirigirse hacia nodos móviles, los cuales, probablemente, se encuentren desconectados del sistema. El mecanismo de trabajo propuesto es el siguiente:

- Para el primer caso, un agente móvil por cada nodo a actualizar, el agente móvil debe esperar en el nodo base por la conexión del nodo móvil al cual se debe dirigir para la actualización.

- En la segunda hipótesis planteada, puede ocurrir que un nodo del itinerario resulte ser un nodo móvil desconectado. En ese caso, el agente móvil estará dotado de la suficiente "inteligencia" como para detectar esta situación e intentar continuar con el camino, dejando pendiente dicho nodo. Esto puede llevar a que varios puntos del recorrido se "salteen" temporariamente hasta que se conecten a la red.

La selección del esquema Lazy Master como modelo de actualización de réplicas no es la mejor solución para un entorno con agentes móviles. En nuestra caso, la elección se debe a que con las suposiciones de base realizadas, se convierte en un esquema más sencillo para implementar como primera aproximación a la solución del problema.

Al suponer que las copias maestras de datos se instalan en nodos base, las actualizaciones generadas, tanto por otros nodos base como por nodos móviles, se procesan secuencialmente (por orden de llegada). Una vez producido el cambio, en caso que la transacción se cometa, las actualizaciones de las otras réplicas secundarias se van a realizar, esto es, no existe la posibilidad de fallo sobre una transacción generada por un nodo base con una copia maestra del objeto que indique actualizar.

Conclusiones y trabajo futuro

Los esquemas Eager no son utilizables para esquemas de actualización de réplicas con agentes móviles. Los esquemas Lazy se adaptan mejor en aspectos como seguridad y performance, además son implementables aún utilizando agentes móviles. El esquema Lazy Master tiene una implementación más sencilla, pero no es adecuado que un nodo móvil contenga una copia maestra.

Por último, el esquema Lazy Group para actualización de réplicas se adapta mejor al problema de utilizar agentes móviles. Las experiencias posteriores están direccionadas hacia este punto, si bien este esquema necesita un algoritmo de reconciliación en algunas circunstancias.

El trabajo futuro está orientado a generar un modelo donde la situación de base planteada se acerque más a la realidad. No es posible suponer, bajo un esquema real, un procesamiento secuencial de transacciones, las mismas deben ser procesadas de acuerdo a su momento de generación. Además, las situaciones de conflicto que se plantean en un entorno con nodos móviles deben ser consideradas.

Se planteó en el artículo la posibilidad que dos transacciones entren en conflicto y se estudió, en líneas generales, la probabilidad de ocurrencia de este caso. En un esquema real, este punto de fallos debe ser tratado y debe, por consiguiente, plantearse un algoritmo de reconciliación para la solución del problema.

Bibliografía

[GRAY 96] *The Dangers of Replications and a Solution*. Gray Jim, Hellian Pat, O'Neil Patrick, Shasha Dennis. SIGMOD 96. Montreal Canada. ACM 0-89791-794-4/96/0006

[BERNSTEIN 87] *Concurrency Control and Recovery in Database Systems.* Bernstein P., Hadzilacos V., Goodman N. Addison Wesley Readings 1987.

[BURETTA 97] *Data Replication Tools and Techniques for Managing Distributed Information*. Buretta Marie. John Wiley & Sons, Inc. 1997.

[HOLLIDAY 2002] *Disconnection Modes for Mobile Databases.* Joanne Holliday, Divakant Agrawal, Amr El Abbady. Wirelles Neworks 8, pp 391-402. Kluwer Academic Publishers. 2002

[LADIN 92] *Providing High Availability Using Lazy Replication*. Rivka Ladin, Barbara Liskov, Sanjay Ghemawat, Liuba Shrira. ACM Transactions on Computer Systems, Vol 10, No. 4, November 1992

[LEN 01] *Estudio de actualización de réplicas de datos en entornos de Bases de Datos Distribuidas.* Sergio Len, Sebastián Ruscuni, Rodolfo Bertone. Anales: Cacic 2001. Congreso Argentino de Ciencias de la Computación. Neuquen. Octubre 2001. Pp 695-706

[MIATON 98] *Expediencias en el Análisis de Fallas en Bases de Datos Distribuidas.* Ivana Miatón, Sebastián Ruscuni, Rodolfo Bertone, Armando De Giusti. Análes: Cacic 98. Congreso Argentino de Ciencias de la Computación. Neuquen. Octubre 1998. pp 265-276

[PEREZ 98] *Agentes Móviles en Bibliotecas Digitales*. Pérez Lezama, C. V. Tesis Maestría. Ciencias con Especialidad en Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas-Puebla. Mayo. Universidad de las Américas-Puebla. 1998.

[RUSCUNI 00] *Evaluación de Replicación y Consistencia en Bases de Datos Distribuidas*. Sebastián Ruscuni, Rodolfo Bertone. Cacic 2000. Congreso Argentino de Ciencias de la Computación. Ushuaia, Octubre 2000. pp 145-158

[SANCHEZ 97] *A taxonomy of agents.* Sanchez, J. Reporte Técnico ICT-97-l. ICT. Laboraty of Interactive and Cooperative Technologies, Departament of Computer Systems Engineering, Universidad de las Américas-Puebla, México, Enero 1997.

[SILBERSCHATZ 98] *Fundamento de Bases de Datos.* Tercera Edición. Silberschatz Abraham, Korth Henry, Sudarshan S. Mc Graw Hill. 1998